Petascale Software Infrastructure Risks and Mitigation Strategies

Contributors in Alphabetical Order: Jonathan Carter, Tony Drummond, Parry Husbands, Paul Hargrove, Bill Kramer, Osni Marques, Esmond Ng, Lenny Oliker, John Shalf, David Skinner, Kathy Yelick,

2/25/2006

In the next 5 years, the DOE expects to build systems that approach a petaflop in scale. In the near term (2 years), DOE will have several "near-petaflops" systems that are 15 to 35% of a petaflop scale systems with a petaflop "peak" system expected in late 2008. In 2011 we expect to see the first systems capable of a petaflop sustained performance on some subset of applications. A common feature of these systems, as evidenced by precursors such as the Cray XT3 or the IBM BlueGene/L, is that they rely on an unprecedented degree of concurrency, which puts stress on every aspect of HPC system design. Such complex systems will likely break current "best practices" for fault resilience, I/O scaling, debugging, and even raise fundamental questions about languages and application programming models. It is important that potential problems are anticipated far enough in advance that they can be addressed in time to ensure that DOE's investment in hardware technology is not wasted. Our response is organized as follows.

- **Section 1** provides a general overview of computer architecture trends that motivate software requirements considered in the subsequent sections.
- **Section 2** provides a high level answer to the question *"What software is on the critical path to make the systems work.* We broadened the response to include both hardware and software issues because the two are so intricately entwined on systems of this scale. We also differentiate near-term (2007-2008) challenges for near-petascale systems from those that we anticipate in the long term when DOE and NSF expect to field systems capable of a sustained petaflop. (2011 and beyond).
- **Section 3** describes a set of codes that provide good representation of the application requirements of the broader DOE scientific community. The success of these codes is bellwether for the overall success of these computing platforms for DOE scientific applications

This document will raise concerns about problem areas of our software infrastructure and points to some research direction that have been targeting gaps in our software stack. However, merely funding the research to develop solutions will not address these problems from a computing center's operational standpoint. Close interaction with the vendors is an absolutely critical component to successfully addressing the concerns outlined in this document.

1 Computer Architecture at the Petaflop Scale

Before we delve into software issues, it is probably useful to review recent trends in computer architecture to get a sketch of some likely features of petaflop scale computing architecture. (eg. what will a petaflop machine look like). One important discontinuity that has developed in system architecture is motivated by changes in device physics below the 90nm scale. The changes at the chip level create a cascade of design choices that affect every aspect of system design and result in some of the most daunting challenges for software design on future petaflop systems.

Figure 1 shows the improvements in processor performance as measured by the SPEC benchmark over the period from 1975 to present. Since 1986 performance has improved by 52 percent per year with remarkable consistency. During that period, as process geometries scaled according to Moore's law, the supply voltages were kept constant in order to allow manufacturers to increase clock-speeds. This approach, known as fixed-voltage scaling[1] fed the relentless increases in CPU clock-rates over the past decade and a half. However, below the 90nm scale for silicon lithography, this technique began to hit its limits as power density has become the dominant constraint in the design of processing elements[2]. The result has been a stall in clock frequency that is reflected in the flattening of the performance growth rates starting in 2002. In 2006, individual processor cores are nearly a factor of three slower than if progress had continued at the historical rate of the preceding decade. The mainstream microprocessor industry has responded by halting further improvements in clock frequency and increasing the number of cores on the chip. Patterson and Hennessy[3] estimate the number of cores per chip is likely to double every 18 months henceforth.

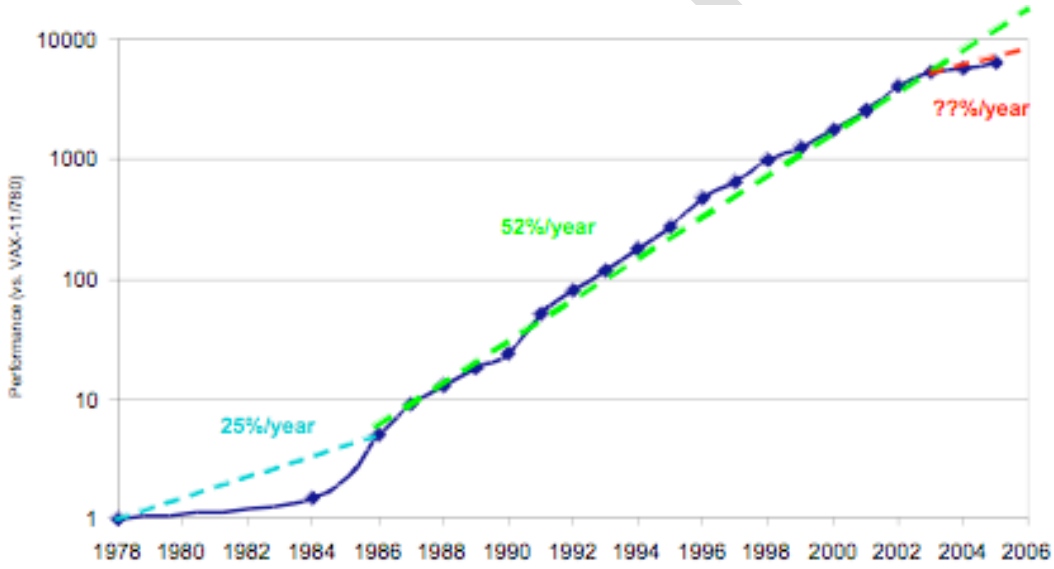

Figure 1: This graph was provided courtesy of David Patterson from Patterson & Hennessy, "Computer Architecture: A Quantitative Approach; forth edition".

The recent trends in the microprocessor industry have important ramifications to the design of petaflop-scale computing systems. As the pace of processor clock rate improvements continues to slow, the path towards realizing petascale computing is increasingly dependent on scaling up the number of processors to

[1] Borivoje Nikolic, Jan M. Rabaey, Anantha Chandrakasan, "Integrated Circuits, a Design Perspective." Prentice Hall, Second Edition, 2003.

[2] S. Borkar, "Design challenges of technology scaling," IEEE Micro, 19(4):23--29, Jul-Aug 1999.

[3] J. L. Hennessy, D.A. Patterson, "Computer Architecture: A Quantitative Approach; fourth edition," Morgan Kaufmann, San Francisco, 2006.

unprecedented concurrencies. This is leading to reconsideration of interconnect design, memory balance, and I/O system design that will have dramatic consequences for the design of future HPC applications and algorithms.

Projections: Given current trends, petaflop scale systems in 2011

- Systems will contain between 200,000 and 1,500,00 processors. (50k-200k "sockets"). Each "socket" in the system will be a chip that contains multiple cores. (multicore)
- In 2011 these multicore chips will contain between 8 and 32 conventional processor cores per chip with some extreme cases that contain hundreds of cores ("manycore"). Consider that the Cisco CRS-1 router currently employs a chip containing 188 processor cores using conventional 90nm process technology. (so "1000 cores on a chip" is not as far away as one might expect)
- As microprocessor manufacturers move their design targets from peak-clock-rate to reducing power consumption and packing more cores per chip, there will be a commensurate trend towards simplification of the core design. This is already evidenced by the architecture of the Pentium-M derived Intel Core-Duo processors that sport pipelines that are half the length of its predecessor, the Pentium4. The trend towards simpler processor cores will likely simplify performance tuning, but will also result in lower sustained performance per core as out-of-order instruction processing is dropped in favor of smaller, less-complex less-power-hungry in-order core designs.
- As the number of cores per socket increase, memory will become proportionally more expensive and power hungry in comparison to the processors. Consequently, cost and power-efficiency considerations will push memory balance (in terms of the quantity of memory put on each node) from the current nominal level of 0.5 bytes of DRAM memory per peak flop, down below 0.1 bytes/flop (possibly even less than 0.02 bytes/flop).
- Cost scaling issues will gradually cause fully-connected interconnect topologies such as the fat-tree and crossbar to be gradually supplanted at the high-end by lower-degree interconnects such as the n-dimensional torii, meshes or alternative approaches such as hierarchical fully-connected graphs. The consequence will be that HPC software designers must take interconnect topology and associated increased non-uniformity in bandwidth and latency into account for both algorithm design and job mapping. At this point in time, the interconnect topology is typically ignored by mainstream code and algorithm implementations).

These trends will have a dramatic effect on software and algorithm design for such systems. The requirements of the software infrastructure that are necessary to accommodate these architectural trends will be considered in more detail in subsequent sections.

2 Software Needs for Petascale and Near-Petascale Computing

In the sections below we characterize the potential deficiencies in system and application software that will exist in the 2007 timeframe for the three HEC architectures of interest to the Office of Science. Section 1.1 summarizes the major issues. The following section 1.2 looks at longer-term issues that must be addressed before petaflop systems can reach a usable state.

2.1 Critical issues for Near-Petascale (circa 2007-2008)

The following issues lie on the critical path for success of near petaflops systems in the near term. We believe that targeted funding in these areas can provide dramatic short-term pay-offs in terms of functionality and efficiency for coming systems.

- **Fault Tolerance**: The emerging crop of near-petascale HPC systems contain an order of magnitude more components than current generation systems. Current software and hardware approaches are inadequate to the task of managing and containing the failure modes that are likely to exist in such complex systems. Fault tolerance is primarily limited by the fragile software environment. For example, once a XT3 router chip fails, the torus can not dynamically change routing. More importantly the router, once fixed, cannot be returned to service without a full system reboot[4]. The BG/L presents similar issues. Hence, the system resources slowly decay until enough nodes are off line that system managers shut down and reboot.

  To address these issues, SW reliability has to be improved. The majority of system wide outages at NERSC are due to software failures.

  Recommendation: *A fundamental review of system software is needed with the goal of reducing complexity and increasing reliability. Ways to proactively judge software reliability will be critical to deciding where resources should be deployed. Attention to fault tolerance in future software/hardware design must rise to a top priority in order to ensure the success of systems that are 10x larger than today's implementations.*

- **MPI and Support for Legacy Programming Models**: Efficient MPI implementations are necessary at increased size. The workload at NERSC and other labs show that majority of the Office of Science workload can work effectively at the scale of 512 to 4096 processors using MPI. The systems in 2007 will have four times the number of CPUs. The current investment in science codes has to be preserved, so MPI will have to operate effectively at the scale of 10,000-40,000 tasks. Certainly the MPI implementation on BG and the XT3 can scale this far, but the IBM SP and OpenMPI implementations are currently not capable of scaling to the hundreds-of-thousands of cores expected in petaflop scale systems.

  Recommendation: *Support for tools and optimizations of legacy programming models must continue even on petaflop scale systems. Even if all new HPC software development were to shift exclusively to advanced parallel programming languages, it will still take more than a decade to shift the existing HPC software infrastructure over to the new programming methods.*

- **Public interfaces for lightweight communication:** As the number of processors grows and the relative cost of the network increases, applications that are communication-intensive must be optimized to make effect use of network resources. Hardware support such as Remote Direct Memory Access (RDMA) offer opportunities to overlap communication with computation using lightweight one-sided communication, but interfaces below MPI are often kept proprietary. Hardware issues, such as lack of cache-coherence between network processors and compute processors within a node, also interfere with the ability to use RDMA. The use of unmodified commodity Operating Systems like Linux can also add overhead due to virtual memory

[4] Note that the XT3 cannot do hot swaps either, so an entire rack needs to be powered down, losing 96 nodes.

management and other services. Benchmarking efforts (e.g. HPC Challenge GUPS benchmark) on the BlueGene/L machine have utilized non-public network interfaces. The LBNL/Berkeley UPC group and PNNL ARMCI groups have demonstrated the value of RDMA-based benchmarks in both microbenchmarks and application level algorithms[5].

Recommendation: Public lightweight communication interfaces are needed to optimize public domain implementations of MPI as well as Global Address Space models (below) in order to expose the best available network performance on each machine.

- **Global Address Space Models**: While we believe support for legacy programming models such as MPI and OpenMP is important, the delivery of efficient implementations of current-generation Global Address Space models like UPC, Co-Array Fortran (CAF) and Global Arrays (GA) provide a user level programming abstraction for the efficient one-sided communication described above. The language-based models offer an advantage over libraries, because they expose parallel constructs to automated optimization and reorganization by the compiler. Otherwise, the semantic meaning of API calls (such as MPI) are opaque to the compiler's optimizer and therefore cannot be optimized in any fashion by the compiler.

  *Recommendation: Global memory languages are poised to play an important role **now** (in the near-petascale time range) for enabling effective use of petaflop-scale systems. UPC compilers are available on every major HPC system architecture, from PC clusters to the X1e, including prototypes for the XT3 and BG/L. LBNL is partnering with Cray and Intrepid to provide optimized UPC support for the XT3, and IBM has an internal "research" compiler for UPC on BG/L. CAF has a portable open-source compiler from Rice, and GA is available from PNNL. These languages have nearly achieved the level of ubiquity necessary to change the software ecosystem.*

- **Stable Parallel Filesystems.** Current global, parallel file systems are able to run effectively at the scale of 2,000 clients. There are no other practical solutions for vendor-supported multi-architecture parallel cluster filesystems in the near term aside from GPFS and Lustre. Lustre has shown significant reliability issues, but otherwise demonstrates effective scaling for a number of clients on the BG/L system at LLNL. GPFS shows moderate scaling, good performance and reliability for concurrencies up to 2,000 clients, but is only just starting to be assessed on near-petascale architectures. GPFS performance and reliability remains relatively unexplored at concurrencies in excess of 2,000 clients. If the current implementations are just scaled to the number of compute nodes in the pre-petaflop/s systems, it is likely significant scaling issues will be encountered -- even show-stopping problems. Several solutions need to be explored, including redesign and the implementation of "lightweight" interfaces on compute nodes.

  Recommendation: The filesystem problems identified in this section cannot be solved by the DOE Labs alone -- even if the software is open-sourced. Any credible solution requires close collaboration between the labs and the two primary vendors who have developed scalable parallel I/O system technologies that is supported by targeted funding to proactively target these issues.

- **Numerical Libraries:** Additional focus is required to ensure the scalability of parallel libraries. Much of the current activity in performance tuning has focused on single processor performance. Scalar performance will continue to be important, but higher level performance issues such as communication performance, load imbalance, synchronization overhead will dominate at high concurrencies. The level of sophistication required for petaflop-capable parallel libraries is dramatically greater than can be derived from current practical experience with existing lower-concurrency implementations on comparatively flat interconnect topologies. Funding of advanced library development that **leads** rather than lags the deployment of these systems will be essential to ensure the systems are used effectively for scientific applications once they come online. Large scale facilities will need to work with library and tool developers to assemble "software testbeds" that will be used to adequately test new platforms and document the results, and systematically

[5] See http://gasnet.cs.berkeley.edu/performance for microbenchmark numbers and "***Optimizing Bandwidth Limited Problems Using One-Sided Communication and Overlap,"*** C. Bell, D. Bonachea, R. Nishtala, K. Yelick. 20th International Parallel & Distributed Processing Symposium (IPDPS), 2006

accumulate a list of requirements and expertise on the software and platforms were it runs. Library and tool developers should be among the community of "early" pre-production users on the near petascale systems and should aggressively use this access to port, test, and with other application-focused early users to uncover problems at the level of the tools (like numerical tool implementations, precision, etc. . .) and applications codes. Early detection of these (through consistent validations) can make the science community.

Recommendation: Both users and library developers have underscored the need for robust software testbeds that allow them to proactively port their software to new platforms using smaller-scale early-release versions of candidate HPC system hardware. A program that enables centers to purchase non-production early/prototype hardware could play an important role in establishing advanced software testbeds that prepare our software infrastructure for eventual full-scale implementations based on the same system architecture.

- **Debugging**: There is no debugging solution for systems of this scale. Totalview does not work for users above 1,000 tasks, and only works on one near-petaflop-scale system beyond 1,000 nodes at all. Further more, the Etnus cost model is extremely expensive at scale given their pricing model is based on the number of CPUs in the system. Continuing on the current direction for development for this software technology is untenable.

  Recommendation: In the short term, it will be necessary to push vendors such as Etnus to extend the capabilities of totalview to support larger-scale platforms, but the costs may be impractical. Yet it may be impossible to assimilate all of the information present in a conventional debugger's view of the program state given the massive concurrency of these systems. Ultimately this issue requires renewed exploration of alternative approaches to debugging at massive concurrency. Targeted funding is required to reinvigorate the research pipeline in order to spawn new ideas in this area. This point is explored in more depth in section 1.2 as a long-term issue.

- **Checkpoint/Restart**: Operationally C/R provides many advantages that make large system more effective By implementing C/R on both the T3e and the IBM SP, NERSC increased its ability "to run the right job at the right time," to meet the needs of long running jobs while still providing good throughput for the general workload, and reducing the impact of system shutdowns. In principle, C/R is feasible at scale on all the systems but currently that capability is only available on the Power Architecture. Linux C/R, developed through the DOE FastOS effort, may find its way into the XT3 and BG/L compute nodes if/when they moved from microkernels to stripped down Linux kernels. Meiosys is another possible implementation of C/R being considered by at least one vendor.

  Recommendation: While this move presents an opportunity for the Linux C/R efforts, it will require software integration and support to ensure production quality services at this scale. It will require renewed support for vendor/laboratory collaborations to bring this work to fruition.

The following problems are lower priority. They will greatly affect the usability and efficiency of the system, but will not necessarily be "show stoppers" to making systems work for production quality science. Systems without these functions merely waste DOE effort invested in these architectures through lower delivered system efficiency.

- **Changing Memory Balance**. Memory is one of the most expensive and power-hungry components of systems. For the past 15 years, systems have consistently hovered around a byte-per-flop ratio of 0.5 bytes per peak FLOP delivered by the system despite the lack of firm scientific grounding for this ratio. Historically it has been associated with Amdahl's heuristic design point for mainframes that called for one byte of memory per instruction/sec[6] processing performance. As systems move to having thousands to tens of thousands nodes, it may not practical to continue the current memory balance on future systems and still have enough funding to reach the computational goals. It is unclear to what extent this design trade-off would impact the design of future applications and numerical algorithms.

[6] Note – this is a non-floating point measure_

The scientific computing community needs to develop the appropriate framework to address this issue in a rational and scientific manner. The process can be informed through the in-depth analysis of algorithm such as the work by supported by the MICS office through base program funding and through SciDAC efforts such as PERCS. These efforts can provide the ultimate answer to this question if it is identified as an important issue, but someone must be chartered to explicitly investigate the B/F ratio for the Office of Science workload.

- **Efficient Job Migration and Torus packing**: In order to reduce performance variability jobs must be given topologically contiguous sections of the 3D torus topology. As jobs are scheduled and retired, gaps develop in the torus that must be removed by migrating running jobs to repack the torus. Such migrations require many of the same capabilities of OS-initiated checkpoint/restart, but the job state need not be written to and from disk. Integrating efficient job migration and job packing algorithms with the batch subsystem will be important for efficient utilization of these system architectures. NERSC implemented such capabilities for the T3E/900 system (mcurie).

  Recommendation: Recreating this capability requires much of the C/R capabilities as a minimum requirement, and must be complemented by an open/modular resource management system that can be modified to direct the job suspension and process migration facilities to effect job migration in response to current job scheduling conditions. The XT3 will present a problem in this regard due to lack of explicit control of job layout in the current system software implementation.

- **Interconnect Topology Awareness / Job Mapping:** Because of their lower-degree (torus) interconnect topology, it is essential for application codes to take topology into account (job mapping). For example, BG/L requires very complex job mappings for anything except the simplest of problems[7]. Currently users have little information to select job mappings other than the canonical mapping and facilities such as the MPI topology directives are not well exploited by current software stacks. There are few tools to assist with selecting a better mapping to the interconnect topology much less automatically generating map files for the user (that is left as an exercise for the user to write, if possible, scripts to generate reasonable mappings). Currently available monitoring tools do not provide useful advice on how to improve the job mapping or are not practical for use at full-system-scale.

  *Recommendation: Funding to develop lightweight, non-invasive systems for collecting communication & performance data from the jobs while they run at production scale will be essential to make more efficient use of these resources. Such performance monitoring frameworks must be non-invasive enough that they can be used for **all** jobs run on the system rather than as an offline performance optimization/tuning step. Combining these capabilities with the facilities to re-pack jobs on the interconnection network will greatly improve resource utilization efficiency.*

- **Parallel I/O:** MPI I/O may be insufficient at this scale unless POSIX semantics can be relaxed. Some form of async I/O or other support for relaxed POSIX compliance will be critical to scalable I/O performance. This capability must be exposed efficiently via mainstream I/O libraries like ParallelNetCDF or parallelHDF5. This is not a show-stopper given many users continue to follow the practice of writing one file per processor. The result of such user behavior is overblown storage system performance requirements (such as the 30k file creates/sec required by DARPA HPCS) that will otherwise increase costs and reduce the efficiency of archival storage systems such as HPCS.

  Recommendation: DOE labs can help by continuing R&D of parallel file formats and libraries such as MPI-I/O, more efficient parallel-I/O alternatives such as server-directed I/O, expanding the effectiveness and portability of parallel global file systems that allow parallel writes to single files in an efficient manner, and refining a strategy for relaxed POSIX I/O. The latter requires close collaboration with industry and some organized effort to promote standardization of the approach across multiple platforms. Refinements in the lower-level parallel I/O strategy must be

[7]Almasi, Bhanot, Gara, Gupta, Sexton, Walkup, Bulatov, Cook, de Supinski, Glosli, Grennough, Gygi, Kubota, Louis, Spelce, Streitz, Williams, Yates, Archer, Moreira, Rendelman: Scaling physics and material science applications on a massively parallel Blue Gene/L system. ICS 2005: 246-252.

complemented by comparable funding for the incorporation of those advances into high-level file storage organization such as pHDF5 and parallelNetCDF.

- **Visualization and Data Analysis Infrastructure:** The ASC VIEWS program has provided a good set of baseline tools to attack this problem such as LLNL VisIT. However, scientists require constant evolution in these baseline tools -- tailoring their capabilities to meet the specialized visualization and data analysis requirements of their scientific domain. Tool development for data analysis and visualization is a continuous optimization process, requiring close collaboration with the domain scientists. VisIT offers a substrate to build upon that meets baseline needs, but such tools do not fix or extend itself. Failure to consider the requirements of data analysis (both hardware and software resource) **together** with the large-scale system procurement will render any such system ineffective.

  Recommendation: DOE needs to preserve the investment in data analysis software infrastructure by continuing to fund collaborations between the providers of tools such as VisIT and Paraview, and the application stake-holders.

2.2 Critical Software Issues for Petascale Systems (circa 2011)

_____ 2.2.1 Fault Tolerance

The emerging set of near-petascale systems present a two order of magnitude leap in scale over typical systems today, in terms of hardware components (total number of disks, processors, and DRAM sticks), which is causing a critical gap to open up in fault management of these systems. Most of the systems have hardware designs that include many features that improve fault tolerance, but critical failure modes persist. Currently, systems software components for large-scale machines remain largely independent in their fault awareness and notification strategies. Software implementations remain fragile and do not support many of the hardware RAS (Reliability Availability Serviceability) features, so operationally, there is limited benefit for the hardware features. Faults can arise not just from the hardware but also from the OS, middleware, and application levels. Failures typically get reported using rudimentary error conditions such as "job failed," that provide little indication of the root cause of the failure. Given the overwhelming complexity of emerging petascale HPC applications, such opaque failure modes may well render the system unusable. Moreover, the multiple layers of software between application and computer have little to no opportunity to report, avoid, or correct the issue.

Fault Tolerance: As mentioned above, the work in hardware fault tolerance is being limited by the fragile software. System designs that have the interconnect integrated into the node (such as BG/L and the XT3) see situations where individual node failures can impact the performance of the rest of the system, but are difficult to recover from.

Checkpoint Restart: C/R is a primary approach for fault-tolerance. It helps preserve application work and also helps system effectiveness when system shutdowns are needed. System wide checkpoint becomes increasingly important because any solution that attempts to improve on fault-tolerance over CPR is likely to creep in to the programming model. For instance, even in the most likely case of moving to application-initiated C/R, there is a need to annotate the minimum amount of state that must be preserved. Entire system C/R is feasible at scale.

However, the time to do a Checkpoint is dominated by the size of memory and the IO bandwidth. Since bandwidth increases more slowly than memory capacity, the time for checkpoints will continue to increase. Hence, it will become important to have incremental/journaled Checkpoints. This is analogous to full and incremental checkpoints. You only do a full checkpoint for an application occasionally, and you do incremental checkpoint that reduce the amount of IO needed. Restart would be longer – having to read in both the full and incremental checkpoints, but that may be more effective.

It is necessary for the batch system to handle the restart, but this is not possible to explain to a batch system in a concise manner where to locate the restart files and how to invoke the application to perform the restart unless there are some common guidelines to enforce uniformity. If you expect one processor failure every

10 minutes, it is a non-solution if the user must intervene in the restart process. Need to develop uniform software frameworks to support uniform app-level C/R if this approach is ever to be workable.

The problems with extending C/R to future systems suggests a need to build an infrastructure necessary to enable systems to adapt to faults in a *holistic* manner. Such a system would provide common uniform event handling and notification mechanisms for fault-aware libraries and middleware. Applications would need an interface to interface with these capabilities in a more seamless manner.

Recommendations: *Attention to fault tolerance in future software/hardware design must rise to a top priority in order to ensure the success of systems that are 10x larger than today's implementations. The focus should be on detecting deviant behavior rather in addition to catastrophic failure. Whereas current system software error detection mechanisms remain largely independent of the error handling at system and hardware level, a more integrated approach to the detection, propagation, notification, recovery, and even prediction of errors conditions is necessary. Given the realities of component failure rates, applications and underlying system services will increasingly need to adapt to fault conditions and take defensive action to recover from faults rather than die with a cryptic errors when any such fault is detected. Methods like Statistic Learning Theory should be explored to provide very early warning indicators from user behavior and application results.*

More work on alternatives to C/R need to be launched in the research community. Developing methods to proactively measure and predict reliability, particularly for software, is another research area. A holistic view of reliability demands that fault detection, notification, and recovery mechanisms be integrated across both the OS and application domain in a uniform manner in order to enable a more holistic/comprehensive approach to fault resilience.

_____ 2.2.2 Application Coupling / Multiphysics Applications

Petascale computing platforms enable much higher fidelity physical processes to be simulated. The structure of such computations will likely increase the number of applications requiring complex couplers to join simulation components that that employ incompatible domain decompositions or underlying numerical algorithms to simulate a broader array of physical processes. There are many aspects of this problem that are now of interest. As the coupling of models become more complex, it becomes more prohibitive to simply combine two different codes into one but rather the trend has been to develop software paradigms to make them interoperate (like MCT, DCT, CCA or other more specific software frameworks) However, the execution control part of the problem has not been optimally addressed by vendors or the middleware given the complexity of the problem.

Recommendation: DOE needs to develop more robust versions of existing coupling toolkits and ways that one can control the launching of multi-executable that need to exchange model variables, fields or data on-line, which ensures these coupled runs scalably and reliably on massively concurrent systems given the capabilities of such systems are likely to make multi-physics couplers more common in the future.

_____ 2.2.3 Analytics/Visualization/Data Management

As we move towards larger scale systems, the issue of data management (data storage and movement to different data storage resources) will become intimately entwined with every aspect of the data analysis and visualization process. There is a tendency to separate the concerns of data analysis from those of the core computing system – but this approach cannot be sustained on the largest scale systems. The ASC program took pains to ensure visualization/analysis requirements were intimately linked to each new system by suggesting the data analysis resources be provisioned as at least 10% of the scale of the primary computing system. Anecdotal evidence from the scientists served by the ASC program indicate this metric has served them well.

Recommendations: The ASC metric for scaling data analysis resources as a fraction of the size of the primary computing platform should be examined to determine if the ratio makes sense for petaflop scale systems. Likewise, investments in advanced & scalable visualization technology must be preserved through funding of joint development efforts between science groups and visualization technology providers.

Otherwise, the current rewards system of these respective groups makes the natural emergence of effective data analysis solutions unlikely.

2.2.4 Massively Parallel Programming Models

Most applications remain unprepared for the level of concurrency exposed in petascale systems and their precursors such as the XT3 and BG/L. Advanced programming languages will likely play an essential role in exposing/expressing enough concurrency to take advantage of the myriad of processors. The path to scalability is increasing the number of user-visible threads, because hardware techniques such as pipelining and instruction level parallelism have reached their practical limits.

Some significant issues are

Load-imbalance: Problems with any degree of irregularity or load imbalance such as Unstructured Grids (e.g., UMT2k), Block-Structured AMR (e.g., Chombo and SAMRAI), and sparse linear algebra (e.g., SuperLU, MUMPS) exhibit lower scalability with current methods. Application level load imbalance will be exacerbated on massively parallel machines as all processes must wait for the slowest to arrive at synchronization points, and the dynamic load balancing required for some applications require asynchronous background communication or other techniques to hide their cost. Advanced dynamic partitioning software such as Zoltan and graph partitioners such as Metis can play a role in a software strategy to detect and react to load imbalance

Small Messages: Increasing the number of processor in a system, especially in an environment with small memory per processor, will lead to smaller messages. The 3D FFT is an example where message sizes get smaller (send nprocs messages to each processor in 1/nprocs) as concurrency scales up. To do better at spreading communication over a longer period of time to reduce bisection bandwidth requirements, one must employ even smaller messages. The overhead associated with sending each message has a significant impact for small messages that are required to spread out the communication costs as described in the Chen/Iancu/Yelick paper on fine-grained communication optimizations for UPC[8]. Current best practice is to aggregate messages into larger buffers, but that approach increases software complexity and cannot be extrapolated to systems with such massive concurrency. Hardware that provides fast communication with low CPU overhead will allow for more effective use of networking hardware. In addition, these facilities must be exposed to user level software, ideally in a common communication interface.

Topology: Codes with large global communication, like 3D FFT and Particle-in-Cell methods, require extra attention to communication locality and the topology of the underlying interconnection network. Experience with BG/L has already highlighted the important of topology optimizing communication performance, and this is likely to become more significant as machines scale. MPI libraries need to optimize for topology, and machines should expose information to allow application programmer to optimize as well.

MultiThreading: The primary path to exploiting hardware threading currently involves mixed-mode OpenMP + MPI programming (OpenMP to exploit threading on each node and MPI to span the distributed memory domains). However, OpenMP offers a rather limited approach to expressing threading concepts that are completely disjoint from the considerations of the message-passing model. Few programmers are willing to take the additional step of employing 2-level parallelism in this manner. Languages that offer a more natural/readable approach to exposing parallelism are required. Vendors can implement UPC and CAF in this time frame, but they offer only an interim solution to exposing the necessary degree parallelism required to make petaflop scale systems even marginally useable.

Recommendation: Continued development of advanced programming models in close collaboration with vendors and application scientists is essential to ensure that the hardware/software architecture for messaging on future systems will meet the needs of advanced programming models, and the programming

[8] Wei-Yu Chen, Costin Iancu, Katherine A. Yelick: Communication Optimizations for Fine-Grained UPC Applications. IEEE PACT 2005: 267-278.

models meet the needs of the users. Cross-agency and vendor coordination is also essential to ensure ubiquity of emerging programming models.

2.2.5 Libraries:

Efficient libraries are essential for large-scale systems. At the petaflop scale, software complexity is too daunting for each individual applications group to take on alone (unless their application is exceedingly simple). The community will, by necessity, be more reliant on numerical algorithm development and performance tuning work that goes in to library solutions.

The increasing sophistication of computer models together with the complexity of algorithms and computer architectures greatly inhibits progress towards optimized version of an application. In addition, a development team may already be faced with software intricacies that cannot be undertaken by the team alone. Together, all these issues may be even more challenging on large-scale systems.

Libraries can provide a convenient way for incorporating state-of-the-art computational technology into development efforts, with the additional benefit of alleviating many aspects of the development process and enabling testing of different techniques or implementations. While library autotuning might be viable in some cases, overall it is likely to be impractical on large-scale systems since it may require increased validation and testing to make sure the libraries are functioning properly.

Exploitation of Serial Libraries: Have gotten a benefit from calling non-parallel libraries (like FFTW) within parallel applications. However, the degree of sophistication required to achieve scalable performance on mesh interconnects (eg. the torus networks on BG/L and the XT3) far exceed current requirements. This will necessitate more focus on the largest scale

Autotuning is potentially impractical at this scale if the typical exhaustive search is employed. The search space for autotuning is greatly expanded if parallelism is admitted into the search space on the same level with scalar optimizations. This may require a hierarchical approach to autotuning of parallel applications or it may require a limiting the range of autotuning. Complex autotuning at this scale may require increased validation and testing to make sure the libraries are functioning properly.

Recommendation: To identify the main libraries used in simulation codes, provide the resources the early porting of such libraries to emerging large-scale architectures, identify eventual bottlenecks in the libraries and recommend alternative solutions.

We reiterate the importance of providing targeted funding for massively concurrent system libraries and programming abstractions. The development of such libraries must anticipate the arrival of the systems (using smaller scale proxys or early-prototypes to assist in development) rather than spin up after such systems have been delivered. The lead-time required for software development is considerable, so a close interaction between the Research Evaluation and Prototype program and developers of advanced software technology is essential for the success of future systems.

2.2.6 Debugging:

No practical debugging solution for problems scaled beyond thousands of processors. Existing solutions for > 512 processors are inadequate. Most users running about this level do not use debugging tools such as Totalview. There currently is no effort underway to replace Totalview and there is little motivation for Etnus to invent and test for higher scale. No responsive solutions in the pipeline from the research community.

Although some tools like Umpire[9] and MARMOT[10] that are capable of detecting faulty use of MPI functionalities have been developed, it is fair to state that there is no practical debugging solution for

[9] Dynamic Software Testing of MPI Applications with Umpire, J. S. Vetter and B. R. de Supinski. Proc. SC2000: High Performance Networking and Computing Conf. (electronic publication), ACM/IEEE. 2000.

problems scaled beyond thousands of processors. Although users do want to employ debug functionalities on more than 512 processors, existing solutions such as Totalview are inadequate. To the best of our knowledge, there is currently is no effort underway to replace Totalview and there is little motivation for Etnus to invent and test for higher scales. At the same time, there are no responsive solutions in the pipeline from the research community.

Recommendation: Higher-level toolkits are needed to provide an alternative approach debugging large scale codes, that go beyond stepping through code with interactive debuggers, print statements, and time stamps that are cumbersome to look when running on hundreds of processors. There is a need for tools that can replicate scenarios and program failures with user friendly interfaces.

2.2.7 Operating System

OS interference is a well documented problem on systems such as ASCI White and NERSC3. The microkernel approach on the XT3 and BG/L mitigates OS interference problems. However, XT3 and possibly BG/L will be moving to stripped down linux kernels. This has ramifications on reliability/uptime and possibly OS interference on large-scale systems that requires additional planning and attention as we lead up to production system deployment. This is potentially a **very** dangerous and show-stopping consequences if we do not proceed with all deliberate caution and preparation for such a move given the reliability of this approach is almost completely unknown at these scales.

Linux OS RPM management: The complex web of hardware/driver dependencies makes any OS upgrade or modification challenging for systems that use a Linux or UNIX base. Furthermore, while considered solid for desktop systems, the reliability of Linux on massive scale remains questionable. The planned move of the XT3 and BG/L compute nodes from microkernels to Linux implementations presents a high degree of risk – particularly if the Linux community refuses to adopt extensions that might improve the reliability and effectiveness of Linux on large scale systems (as they have done many times in the past).

Virtual Machines and Sub-OS Solutions: Modern OS Virtualization/Virtual Machine Monitors (typically referred to as VMM or Hypervisor) technology is typically employed to enable multiple simultaneous OS instances to run simultaneously on the same hardware[11]. However, the VMM technology has also been applied to eliminate the need for the OS in some contexts where a realtime microkernel may otherwise be considered. For instance, the "Virtual Appliance" approach explored in the Collective project[12] was proposed in response to the phenomenal growth rate in the complexity of OS complexity. Over time OS's accrete objects and API interfaces at a phenomenal rate in order to accommodate all possible use case for the system. For instance, over the space of 10 years, Linux has grown from 2k objects and 40k API interfaces to 12k objects and 710k API interfaces – most of which are unused by the typical application. If we use the "Virtual Appliance" approach, the required APIs for OS services are compiled directly in to the application code (and **only** the APIs that are required). This has the potential to dramatically reduce the footprint of services that would otherwise be part of the OS, whether or not they are used.[13] Such an approach can reduce the OS footprint on compute nodes while offering more complete functionality than is available from a strict microkernel implementation. This approach is used more commonly in the embedded market where there is more sensitivity to cost and software complexity issues. Perhaps the HPC industry can learn from the embedded processor space.

---

[10] Runtime Checking of MPI Applications with MARMOT, B. Kramer, M. S. Muller and M. M. Resch. ParCo 2005, minisymposium on Tools Support for Parallel Programming, Malaga, Spain, Sep. 12-16, 2005.

[11] M. Rosenblum, Tal Garfinkel, " Virtual Machine Monitors: Current Technology and Future Trends," IEEE Computer, May 2005, pp39-47.

[12] R. Chandra et al., "The Collective: A Cache-Based Systems Management Architecture," Proc. Symp. Network Systems Design and Implementation, Usenix, 2005, to appear.

[13] Todd Veldhuizen, "Software Libraries and their Reuse: Entropy, Kolmogorov Complexity, and Zipfs Law," Library-Centric Software Design (LCSD05) workshop, OOPSLA in SanDiego CA, October 16-20, 2005

When the VM is used as the substrate to run the client application on compute nodes of an HPC system, the state associated with software outside of the application thereby facilitating simpler checkpoint-restart capabilities. Finally, whereas microkernel implementations typically involve hundreds of thousands of lines of code, VM's are typically implemented using 5000 or fewer lines of source code – thereby reducing the complexity of the solution as well as the memory footprint of the runtime layer on the compute nodes.

Recommendation: The concurrency of petaflop-scale systems is far beyond the design point of mainstream OS implementations and therefore requires targeted research to identify and correct deficiencies that arise at such massive scale. Research into innovated OS concepts for HPC (or even OS replacements) is essential for future systems and it is critical to keep the research pipeline filled. However, when it comes to successful production HPC systems, the fact remains that the OS is the primary vehicle for vendors to expose all of their native hardware capabilities at the lowest level. As such, vendors have a keen interest in controlling the development cycle for the OS since it is a mission-critical component of their system design. While continued collaboration and technology transfer between the OS research efforts in the DOE are essential to drive future innovations, from an operational standpoint, any production OS issues must be the responsibility of the vendor.

_____ 2.2.8 Profiling and Performance Analysis:

There are tools that have been used successfully at large scale on BG/L, but limits to the level of introspection and understanding of such enormous collections of performance data. Current approaches rely on pie charts, bar-graphs, and tables of the aggregated bulk performance data.

There are tools that have been used successfully to analyze performance data on thousands of processors. In particular, TAU has been used to generate and analyze profiling information of a CFD code on 16 K nodes on LLNL's BG/L. That same data was later replicated four times in order to simulate the profiling of the application on 64 K processors and test the scalability of TAU. The last version of Paraprof, TAU's visualizer, offers 3-D view functionalities (such as triangle mesh plot and scatter plot) that can help users look into performance data obtained on a large number of processors.

Another solution for performance analysis is IPM (Integrated Performance Monitoring) that is discussed in other publications[14]. The primary contribution of IPM is that it presents a lightweight approach to application monitoring. The overheads (< 3% typically) are low enough to enable continuous monitoring and data collection for a system workload. This has enormous benefits for better characterizing the workload at a given center in order to feed back in to the design requirements for future systems.

It remains to be investigated, however, how tools such as the ones mentioned above behave on a wide set of large applications, the overhead resulting from to their use, their performance, and their capability in helping the user make sense of enormous collections of profiling and tracing data.

Recommendation: DOE needs to continue investment into scalable performance monitoring and visualization software. The dramatic increase in system make data collection and intepretation daunting if we attempt to continue with current approaches to performance profiling. (Such concurrency was never conceived of during their design and implementation.) Critical to all of this are methods of visualizing and distilling data into a usable/actionable form.

_____ 2.2.9 Parallel I/O:

Bandwidth: In hardware, bandwidth is a factor of the number of IO servers, the number and capabilities of the controllers, the number of storage units, and the number of connections from disk devices, as well as the ability of the software to manage the control of operations. The software control is the primary area to attack since the other constraints are a matter of buying more hardware.

[14] S.A. Kamil, J. Shalf, L. Oliker, D. Skinner," Understanding Ultra-Scale Application Communication Requirements," IEEE International Symposium on Workload Characterization (IISWC) Austin Texas, October 6-8, 2005. (LBNL-58059) iiswc05.pdf. (http://ipm-hpc.sourceforge.net/)

Reliability of the Underlying Storage Medium: High performance disk systems have a price point well above commodity disks. Much of the cost resides in the controller, which uses embedded software. Other aspects are trying to use disks, such as SCSI, that have lower duty cycle design points – and hence reliability issues. Current filesystem software stacks will be unable to cope with the move to less reliable storage devices, so fault recovery and resilience pathways must be reexamined in the light of this impending hardware constraint.

Metadata Performance: Standard operations such as simply running a cron job that purges old files (clean scratch) will be impractical on a filesystem with hundreds of millions of files.

File Create Performance:. HPCS 30k file creates/sec is considered unreasonable, but may be a practical requirement for these systems unless concurrent file I/O performance problems are addressed.

Recommendations:

Alternatives to MPI/IO such as Server Directed I/O that were developed for the database community should be reconsidered. These approaches were more effective because the data was committed to disk using fences that offer greater opportunities to overlap computation with communication than the imperative approach of MPI-I/O. It is sad that the techniques were overlooked by the HPC community.

Global File Systems: Facility-Wide File Systems (FWFS)[15] *provide consolidated storage for online user data, replacing traditional system-local parallel file systems for home directories, scratch storage, and project storage. While a FWFS are external to all computational systems, it is mounted natively and at high performance. FWFS grow and evolve over time, serving several generations of computational systems.*

The benefits of FWFS to scientific productivity are manifold. By providing a single unified namespace, FWFS will make it easier for users to manage their data across multiple systems. Users will no longer need to keep track of multiple copies of programs and data; they will no longer need to copy data between NERSC systems for pre- and post-processing. Storage utilization will become more efficient through decreased fragmentation. Computational resource utilization will become more efficient as users can more easily run jobs on an appropriate resource. Storage allocations (quotas) will become larger, because they will no longer be fragmented among several systems. FWFS will also provide improved methods of backing up user data that has the potential to mitigate disturbance to users when block backups are run.

Anticipated developments in file system technology will provide further benefits. It is expected that FWFS will provide Hierarchical Storage Management (HSM) functionality, in which data can be automatically migrated to and retrieved from tertiary storage. This will further improve scientific productivity by enabling the user perception of extremely large online disk (very large quotas) and making it unnecessary for users to manually transfer data to and from HPSS. Other possible technology enhancements include enhanced security, wide area access, and tighter integration with grid technologies.

GPFS is the leading candidate because it has performance, scalability, reliability, better security, can be geographically distributed and is interfaced to two different archive systems. Unfortunately, GPFS is not ported to all potential Petascale systems. Lustre is an evolving candidate that currently has no plans for archive integration, presents security concerns and reliability issues.

In the Petascale domain, there will be 10 to 100's of TB data files. Moving them between systems in a facility will be difficult and time consuming. Indeed, some project teams currently devote .5 to 1 FTE just to move files to the right place. Hence it is essential petascale vendors support file systems that integrate well, and at high performance, into facility wide file systems. However, it is important to encourage some level of competition in this area as well, so DOE should fund work in at least two global, parallel file systems.

[15] In FY2005, NERSC began deployment of a Facility-Wide File System (FWFS)

_____ 2.2.10 Network I/O:

Computational switches are based on very large frame (packet) sizes, typically 64 KB, and suffer performance degradation at lower sizes. Local-area networks based on Gigabit Ethernet can run up to 9 KB jumbo frames reliably for onsite mass storage and backups. The 9 KB frame size increases performance compared to standard Ethernet by reducing overhead and CPU load; but compared to 64 KB frames, it has lower performance and higher latency than computational switches. Wide-area networks typically are restricted to 1,500-byte frames because of the need to support millions of simultaneous connections, even though the main protocols, 10 Gigabit Ethernet, SONET and ATM, will pass frames up to 9 KB.

This mismatch in maximum transfer unit (MTU) sizes, and more importantly the performance degradation that is associated with running small MTUs, makes it difficult to effectively integrate a massively parallel computer system into any networked environment. Furthermore, internal interconnects do not run IP but instead use switch sprciifc protocols such as LAPI and Portals. The two major approaches used to date have been to add an external interface into each node or to turn one or more computational nodes into gateway routers and have external traffic flow across the internal switch to these gateways.

Both of these approaches have major weaknesses. Petascale computing will require thousands to tens of thousands of nodes[16]. Adding an external gigabit interface to every compute node for external connectivity to data storage systems and other computational resources is not practical, nor will it allow single stream performance to increase above current levels, even though this has the best chance for meeting large aggregate bandwidth requirements. Using a compute node that has a connection to the internal switch fabric as well as multiple bonded Gigabit Ethernet interfaces or 10 Gb/s Ethernet as a router is logistically easier and potentially has better single stream performance, but aggregate performance will suffer, especially coupled with small MTU traffic on the computational switch. Further, not all systems have nodes that can drive 10 Gigabits at near optimal rates. Also, a compute node that has to run packets through its IP stack to divide the traffic into packets, generate packet headers, and perform flow and congestion control will never be able to keep up with the fastest switches and routers that just store and forward with all decisions in application-specific integrated circuit (ASIC).

One possible solution would be a Layer-7 router with a computational switch interface that could do the bridging, MTU repackaging, and load balancing. This would be directly connected onto the vendors internal interconnect. No potential Petascale vendors has this in plan. A more workable solution would be to modify a compute node to act like a high performance router. Most of the hardware components for modifying a compute node already exist or would be simple to create. A computational switch, such as a Federation SMA3 adapter, and a 10 Gigabit Ethernet card, both with sufficient field-programmable gate array (FPGA) space to offload the IP protocol, would limit the compute nodes duties to setting up remote direct memory access (RDMA) between the two interfaces. Frame fragmentation and coalescing are often done through TCP proxies but should be programmable in a reasonable amount of FPGA space, thus reducing the adverse impact of widely disparate MTUs.

Recommendation: Network transport and I/O issues need to be considered as part of the Petascale software stack. While a focus on application-requirements is key to a usable system, other supporting software infrastructure such as the network/IP stack must be considered as well in order to deliver a fully functional system!

[16] C. William McCurdy, Rick Stevens, Horst Simon, et al., "Creating Science-Driven Computer Architecture: A New Path to Scientific Leadership," Lawrence Berkeley National Laboratory report LBNL/PUB-5483 (2002), http://www.nersc.gov/news/ArchDevProposal.5.01.pdf.

3 Bellwether Ultrascale Applications

This section outlines a set of codes that provide good coverage application requirements for the DOE Scientific Community, both in terms of application areas and the key numerical algorithms. If the application codes outlined in Table 1 are able to scale to large concurrencies successfully on petaflops precursors such as the XT3 and BG/L, then there is an opportunity to apply the lessons learned to the broader DOE community and thereby ensure the success of petaflop scale systems when they emerge. More information about these applications can be found in papers written by Leonid Oliker et al. (http://crd.lbl.gov/~oliker).

Name	Lines of Code	Discipline	Problem and Method	Structure
MADCAP	5000	Cosmology	CMB analysis via Newton-Raphson using ScaLAPACK	Dense Matrix
Cactus	84,000	Astrophysics	Einstein's Theory of GR via Finite Differencing	Structured Grid
LBMHD	1,500	Plasma Physics	Magneto-Hydrodynamics via Lattice Boltzmann	Lattice
FVCAM	200,000+	Climate Modeling	Atmospheric Circulation via Finite Volume	Structured Grid
GTC	5,000	Magnetic Fusion	Vlasov-Poisson Equation via Particle in Cell	Particle/Grid (PIC)
SuperLU	42,000	Linear Algebra	Sparse Iterative Solver using LU Decomposition	Sparse Matrix
PMEMD	37,000	Life Sciences	Molecular Dynamics via Particle Mesh Ewald	Particle
PARATEC	50,000	Material Science	Material Science Density Functional Theory via FFT	Spectral/3D-FFT + Dense Matrix
SuperNova	200,000+	Combustion	Rayleigh-Taylor via Adaptive Mesh Refinement	SAMR
GAMESS	500,000	Chemistry	Density Functional Theory	
Milc		Quantum Chromodynamics	Conjugate gradient algorithm for Physics on a 4D lattice	Sparse

Table 1 Scientific application suite for evaluation study representing a broad spectrum of numerical methods

3.1 Astrophysics: Cactus

One of the most challenging problems in astrophysics is the numerical solution of Einstein's equations following from the Theory of General Relativity (GR): a set of coupled nonlinear hyperbolic and elliptic equations containing thousands of terms when fully expanded. The Cactus Computational ToolKit (CCTK) is designed to evolve Einstein's equations stably in 3D on supercomputers to simulate astrophysical phenomena with high gravitational fluxes – such as the collision of two black holes and the gravitational waves radiating from that event. The standard MPI driver for Cactus solves the PDE on a local grid section and then updates the values at the ghost zones by exchanging data on the faces of its topological neighbors in the domain decomposition. The update pattern employed by Cactus is typical of a broad class of PDE solvers that use stencils on structured grids to iterate towards a solution. Therefore, the lessons learned

from optimizing Cactus for the XT3 and BG/L are entirely transferable to many important codes in the DOE community.

3.2 Plasma Physics: LBMHD

Lattice Boltzmann methods (LBM) have proved a good alternative to conventional numerical approaches for simulating fluid flows and modeling physics in fluids. The basic idea of the LBM is to develop a simplified kinetic model that incorporates the essential physics, and reproduces correct macroscopic averaged properties. Recently, several groups have applied the LBM to the problem of magneto-hydrodynamics (MHD) with promising results. LBMHD simulates the behavior of a two-dimensional conducting fluid evolving from simple initial conditions and decaying to form current sheets. The computational structure of LBMHD uses a 2D spatial grid coupled to an octagonal streaming lattice and block distributed over a 2D processor grid.

Due to the relatively simplicity of LBMHD's computational structure, combined with the significant disparity between scalar and vector performance demonstrated in our work, this code has been chosen as an application benchmark for the DOD HPCS evaluation effort. We plan to continue working closely with the HPCS community in generating full-scale benchmark applications and disseminating our findings to the HPC community at-large.

3.3 Magnetic Fusion: GTC

The Gyrokinetic Toroidal Code (GTC) is a 3D particle-in-cell (PIC) application developed at the Princeton Plasma Physics Laboratory to study turbulent transport in magnetic confinement fusion. GTC is currently the flagship SciDAC fusion microturbulence code. Turbulence is believed to be the main mechanism by which energy and particles are transported away from the hot plasma core in fusion experiments with magnetic toroidal devices. An in-depth understanding of this process is of utmost importance for the design of future experiments since their performance and operation costs are directly linked to energy losses. GTC simulations of plasma microturbulence in a magnetic fusion device show elongated "finger-like" structures are turbulent eddies in the electrostatic potential that act as energy and particle transport channels. This type of calculation helped to shed light on "anomalous" energy transport that was observed in real experiments.

This code could potentially be scaled up to very large numbers of processors — we therefore plan to conduct an extensive analysis of the tradeoffs between using a very high level of coarse-grained parallelism (as found in BG/L) and utilizing a relatively small number of processors that leverage fine-grained vector parallelism. A new data-decomposition scheme for GTC may, for the first time, enable a breakthrough of the Teraflop barrier and prepare the way for GTC simulations on near-petaflop platforms like the BG/L and larger XT3 systems. We are working to help establish GTC as an HPCS application benchmark due to the importance of this SciDAC fusion application, and its potential to utilize ultra-scale computational resources.

3.4 Material Science: PARATEC

PARATEC (PARAllel Total Energy Code) performs ab-initio quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set. The pseudopotentials are of the standard norm-conserving variety. Forces can be easily calculated and used to relax the atoms into their equilibrium positions. PARATEC uses an all-band conjugate gradient (CG) approach to solve the Kohn-Sham equations of Density Functional Theory (DFT) and obtain the ground-state electron wavefunctions. PARATEC simulations are used to better understand nuclear magnetic resonance experiments. In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in real space and the remainder in Fourier space using specialized parallel 3D FFTs to transform the wavefunctions.. Due to PARATEC's global communication requirements, architectures with a poor balance between their bisection bandwidth and computational rate will suffer performance degradation at higher concurrencies.

PARATEC is an extremely useful tool in evaluating the balance between computational resources and global interprocessor communication facilities, and will be a critical component of our study in evaluating

the tradeoffs of evolving interconnect designs. Furthermore, a recent survey of NERSC ERCAP requests for materials science applications showed that Density Functional Theory (DFT) codes similar to PARATEC accounted for nearly 80% of all HPC cycles delivered to the Materials science community. Therefore, PARATEC is an excellent proxy to the application requirements of the entire Materials Science community.

3.5 Cosmology: MADCAP

The Cosmic Microwave Background (CMB) is a snapshot of the Universe some 400,000 years after the Big Bang. The pattern of anisotropies in the CMB carries a wealth of information about the fundamental parameters of cosmology. Realizing the extraordinary scientific potential of the CMB requires making precise measurements of the microwave sky temperature over a significant fraction of the sky at very high resolution. Such measurements are made by scanning the sky for as long as possible with a cryogenically cooled telescope and as many microwave detectors as possible. The reduction of the resulting datasets — first to a pixelized sky map, and then to an angular power spectrum — is a serious computational challenge, and one which is only getting worse with increasing dataset sizes, as we try to make ever more precise measurements. It is therefore critical to choose the optimal algorithmic approach and supercomputing platform; one approach is the Microwave Anisotropy Dataset Computational Analysis Package (MADCAP), which has been widely used on a variety of HPC platforms.

The full MADCAP spectral estimator includes a large number of special-case features, from preliminary data checking to marginalization over foreground templates, which dramatically increase the size and complexity of the code without altering its basic operational structure. For simplicity, we are therefore developing a stripped-down version, called MADbench, a lightweight version of the MADCAP expressly designed for benchmarking, which retains the operational complexity and integrated system requirements of the full application. We plan to publicly distribute MADbench, and use our synthetic benchmark generation methodology as a model for creating full-scale portable codes for community-wide evaluation efforts. Additionally, we are in the process of distributing MADbench to the HPCS community, as it combines the potential for ultra-scale parallelism with the challenges of heavy I/O requirements. Several other DOE-supported groups including the FastOS research community have shown interest in in MADbench, and we plan to work closely together to support their research effort. Finally, utilizing MADbench on the latest generation of HEC architectures, will allow us to understand the system balances of various platforms, especially in the context of I/O — a critical issue for future data-intensive analyses.

3.6 Climate Modeling: FVCAM

The Community Atmosphere Model (CAM) is the atmospheric component of the flagship Community Climate System Model (CCSM3.0). Developed at the National Center for Atmospheric Research (NCAR), the CCSM3.0 is extensively used to study climate change. The CAM application is an atmospheric general circulation model (AGCM) and can be run either coupled within CCSM3.0 or in a stand-alone mode driven by prescribed ocean temperatures and sea ice coverage. AGCMs are key tools for weather prediction and climate research. They also require large computing resources: even the largest current supercomputers cannot keep pace with the desired increases in the resolution of these models.

Lenny Oliker's group at LBNL will continue the investigation of FVCAM across a broad spectrum of computing platform using very-large grid configurations and thousands of processors, with the goal of isolating the performance-limiting architectural features. Due to the limited number of coarse-grained domain decompositions, FVCAM represents an example of a code that may not benefit from ultra-parallel systems, but must instead utilize fine-grained parallelism to extract the necessary degree of concurrency to scale up to large systems. Our study will focus on understanding these tradeoffs for the latest HEC platforms.

3.7 Combustion: AMR

A Type Ia supernova explosion likely begins as a nuclear runaway near the center of a carbon-oxygen white dwarf. The outward propagating flame is unstable to the Landau-Darrieus, Rayleigh-Taylor, and Kelvin-Helmholtz instabilities, which serve to accelerate the flame to a large fraction of the speed of sound. At present, the exact mechanism for the subsequent explosion of the star is unknown and investigators have turned to numerical simulation. The SuperNova code, developed by researchers in the Center for Computational Sciences and Engineering at LBNL, is one such code that models the unstable flow processes in reacting, degenerate nuclear matter that occurs in white dwarf stars. SuperNova utlizes Adaptive Mesh Refinement (AMR): a technique for automatically refining regions of the physical domain concentrating computational resources where interesting physics is occurring.

Massive parallelism exacerbates long-standing problems with balancing the need for communication locality with memory balance and load-balancing requirements for AMR codes. Measuring and analyzing these tradeoffs is a critical step in understanding the potential of effectively using massively parallel architectures for future AMR computations. It is commonly believed that adaptive calculations will become a key component of future high-fidelity multi-scale simulations, across of broad spectrum of application domains. However, to date, no AMR performance results at the enormous concurrencies offered at the massive degree of parallelism of petascale systems are currently available to the scientific community. The lessons learned from running AMR codes on such systems will guide system requirements for codes that use more intelligent/targeted use of HPC resources rather than resorting to brute force to meet the needs of multi-scale scientific problems.

3.8 Linear Algebra: SuperLU

An emerging method for scientific computation is the use of sparse matrix methods. We plan to explore this important class of algorithms. Our current candidate is SuperLU. SuperLU is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. The library routines performs an LU decomposition with partial pivoting and triangular system solves through forward and back substitution. The LU factorization routines can handle non-square matrices but the triangular solves are performed only for square matrices. The matrix columns may be preordered (before factorization) either through library or user supplied routines. This preordering for sparsity is completely separate from the factorization. Working precision iterative refinement subroutines are provided for improved backward stability. Routines are also provided to equilibrate the system, estimate the condition number, calculate the relative backward error, and estimate error bounds for the refined solutions. The irregular data access patterns of this code presents a performance challenge for superscalar systems, while the complexity of the control flow is expected to inhibit scalability and exacerbate load-imbalances on massively parallel systems such as BG/L and the XT3. This class of codes is at odds with traditional cache-based architectures, and will allow us to evaluate the tradeoffs between various classes of memory hierarchies. Like AMR, SuperLU offers insight into the requirements of applications that are able to attack larger problems using elegance rather than brute force. The requirements for such applications must be considered as essential for petascale platforms – lest we end up with overly specialized hardware architectures that force us into brute-force numerical approaches.

3.9 Life Sciences: PMEMD

Finally, we are interested in investigating algorithms in the field of molecular dynamics, due to their increasing importance and computational irregularity. One candidate code is PMEMD. PMEMD an application that per- forms molecular dynamics (MD) simulations and minimizations using Particle mesh Ewald molecular dynamics. The force evaluation is performed in an efficiently parallel manner using state of the art numerical and communication methodologies. PMEMD uses a highly asynchronous approach to communication for the purposes of achieving a high degree of parallelism. Variants on the parallel particle mesh calculations in PMEMD are used in several other MD codes. The dynamic nature of this code is at odds inhibits data-parallelism and will present a significant challenge for vector architectures. This application may therefore benefit more from ultra-parallel systems than architectures that depend on fine-grained chip-level parallelism. Our study will investigate these competing tradeoffs.

3.10 CHEMISTRY: GAMESS

GAMESS[17] is a program for *ab initio* molecular quantum chemistry. Briefly, GAMESS can compute SCF wavefunctions ranging from RHF, ROHF, UHF, GVB, and MCSCF. Correlation corrections to these SCF wavefunctions include Configuration Interaction, second order Perturbation Theory, and Coupled-Cluster approaches, as well as the Density Functional Theory approximation. Nuclear gradients are available, for automatic geometry optimization, transition state searches, or reaction path following. Computation of the energy hessian permits prediction of vibrational frequencies, with IR or Raman intensities. Solvent effects may be modeled by the discrete Effective Fragment Potentials, or continuum models such as the Polarizable Continuum Model. Numerous relativistic computations are available, including third order Douglas-Kroll scalar corrections, and various spin-orbit coupling options. The Fragment Molecular Orbital method permits use of many of these sophisticated treatments to be used on very large systems, by dividing the computation into small fragments.

A variety of molecular properties, ranging from simple dipole moments to frequency dependent hyperpolarizabilities may be computed. Many basis sets are stored internally, together with effective core potentials, so all elements up to Radon may be included in molecules. Most computations can be performed using direct techniques, or in parallel on appropriate hardware.

3.11 QCD: Milc

The MILC Code is a set of codes developed by the MIMD Lattice Computation (MILC) collaboration for doing simulations of four dimensional SU(3) lattice gauge theory on MIMD parallel machines. The latest version of this code includes libraries and routines for SU(2) gauge theory as well. The MILC Code is publicly available for research purposes.

[17] General Atomic and Molecular Electronic Structure System" M.W.Schmidt, K.K.Baldridge, J.A.Boatz, S.T.Elbert, M.S.Gordon,J.H.Jensen, S.Koseki, N.Matsunaga, K.A.Nguyen, S.Su, T.L.Windus, M.Dupuis, J.A.Montgomery *J. Comput. Chem.*, 14, 1347-1363(1993).